

---

# **ulid Documentation**

*Release 1.1.0*

**Andrew Hawker**

**Sep 06, 2021**



# CONTENTS

<b>1 Modules</b>	<b>3</b>
<b>Python Module Index</b>	<b>19</b>
<b>Index</b>	<b>21</b>



Universally Unique Lexicographically Sortable Identifier.



## 1.1 ulid/api

Defines the public API of the *ulid* package.

`ulid.api.new()` → *ulid.ulid.ULID*

Create a new *ULID* instance.

The timestamp and randomness values are created from the instance Provider.

**Returns** *ULID* from current timestamp

**Return type** *ULID*

`ulid.api.parse(value: Union[int, float, str, bytes, bytearray, memoryview, uuid.UUID, ulid.ulid.ULID])`

→ *ulid.ulid.ULID*

Create a new *ULID* instance from the given value.

---

**Note:** This method should only be used when the caller is trying to parse a *ULID* from

---

a value when they're unsure what format/primitive type it will be given in.

**Parameters** **value** (*ULIDPrimitive*) – *ULID* value of any supported type

**Returns** *ULID* from value

**Return type** *ULID*

**Raises** **ValueError** – when unable to parse a *ULID* from the value

`ulid.api.create(timestamp: Union[int, float, str, bytes, bytearray, memoryview, datetime.datetime, ulid.ulid.Timestamp, ulid.ulid.ULID], randomness: Union[int, float, str, bytes, bytearray, memoryview, ulid.ulid.Randomness, ulid.ulid.ULID])` → *ulid.ulid.ULID*

Create a new *ULID* instance using the given timestamp and randomness values.

The following types are supported for timestamp values:

- `datetime`
- `int`
- `float`
- `str`
- `memoryview`
- `Timestamp`
- `ULID`

- `bytes`
- `bytearray`

The following types are supported for randomness values:

- `int`
- `float`
- `str`
- `memoryview`
- `Randomness`
- `ULID`
- `bytes`
- `bytearray`

#### Parameters

- **timestamp** (See *docstring for types*) – Unix timestamp in seconds
- **randomness** (See *docstring for types*) – Random bytes

**Returns** ULID using given timestamp and randomness

**Return type** `ULID`

#### Raises

- **ValueError** – when a value is an unsupported type
- **ValueError** – when a value is a string and cannot be Base32 decoded
- **ValueError** – when a value is or was converted to incorrect bit length

`ulid.api.from_bytes` (*value: Union[bytes, bytearray, memoryview]*) → `ulid.ulid.ULID`  
Create a new `ULID` instance from the given `bytes`, `bytearray`, or `memoryview` value.

**Parameters** **value** (`bytes`, `bytearray`, or `memoryview`) – 16 bytes

**Returns** ULID from buffer value

**Return type** `ULID`

**Raises** **ValueError** – when the value is not 16 bytes

`ulid.api.from_int` (*value: int*) → `ulid.ulid.ULID`  
Create a new `ULID` instance from the given `int` value.

**Parameters** **value** (`int`) – 128 bit integer

**Returns** ULID from integer value

**Return type** `ULID`

**Raises** **ValueError** – when the value is not a 128 bit integer

`ulid.api.from_str` (*value: str*) → `ulid.ulid.ULID`  
Create a new `ULID` instance from the given `str` value.

**Parameters** **value** (`str`) – Base32 encoded string

**Returns** ULID from string value



**Return type** *ULID*

**Raises** **ValueError** – when the value is not 26 characters or malformed

`ulid.api.from_uuid` (*value: uuid.UUID*) → *ulid.ulid.ULID*

Create a new *ULID* instance from the given *UUID* value.

**Parameters** **value** (*UUID*) – *UUIDv4* value

**Returns** *ULID* from *UUID* value

**Return type** *ULID*

`ulid.api.from_timestamp` (*timestamp: Union[int, float, str, bytes, bytearray, memoryview, datetime.datetime, ulid.ulid.Timestamp, ulid.ulid.ULID]*) → *ulid.ulid.ULID*

Create a new *ULID* instance using a timestamp value of a supported type.

The following types are supported for timestamp values:

- `datetime`
- `int`
- `float`
- `str`
- `memoryview`
- `Timestamp`
- `ULID`
- `bytes`
- `bytearray`

**Parameters** **timestamp** (*See docstring for types*) – Timestamp in milliseconds

**Returns** *ULID* using given timestamp and new randomness

**Return type** *ULID*

**Raises**

- **ValueError** – when the value is an unsupported type
- **ValueError** – when the value is a string and cannot be Base32 decoded
- **ValueError** – when the value is or was converted to something 48 bits

`ulid.api.from_randomness` (*randomness: Union[int, float, str, bytes, bytearray, memoryview, ulid.ulid.Randomness, ulid.ulid.ULID]*) → *ulid.ulid.ULID*

Create a new *ULID* instance using the given randomness value of a supported type.

The following types are supported for randomness values:

- `int`
- `float`
- `str`
- `memoryview`
- `Randomness`
- `ULID`

- `bytes`
- `bytearray`

**Parameters** `randomness` (See *docstring for types*) – Random bytes

**Returns** ULID using new timestamp and given randomness

**Return type** `ULID`

**Raises**

- `ValueError` – when the value is an unsupported type
- `ValueError` – when the value is a string and cannot be Base32 decoded
- `ValueError` – when the value is or was converted to something 80 bits

**class** `ulid.api.Timestamp` (*buffer: Union[bytes, bytearray, memoryview]*)

Represents the timestamp portion of a ULID.

- Unix time (time since epoch) in milliseconds.
- First 48 bits of ULID when in binary format.
- First 10 characters of ULID when in string format.

**property** `str`

Computes the string value of the timestamp from the underlying `memoryview` in Base32 encoding.

**Returns** Timestamp in Base32 string form.

**Return type** `str`

**Raises** `ValueError` – if underlying `memoryview` cannot be encoded

**property** `timestamp`

Computes the Unix time (seconds since epoch) from its `memoryview`.

**Returns** Timestamp in Unix time (seconds since epoch) form.

**Return type** `float`

**property** `datetime`

Creates a `datetime` instance (assumes UTC) from the Unix time value of the timestamp with millisecond precision.

**Returns** Timestamp in datetime form.

**Return type** `datetime`

**property** `bin`

Computes the binary string value of the underlying `memoryview`.

**Returns** Memory in binary string form

**Return type** `str`

**property** `bytes`

Computes the bytes value of the underlying `memoryview`.

**Returns** Memory in bytes form

**Return type** `bytes`

**property** `float`

Computes the float value of the underlying `memoryview` in big-endian byte order.

**Returns** Bytes in float form.

**Return type** *float*

**property hex**

Computes the hexadecimal string value of the underlying `memoryview`.

**Returns** Memory in hexadecimal string form

**Return type** *str*

**property int**

Computes the integer value of the underlying `memoryview` in big-endian byte order.

**Returns** Bytes in integer form.

**Return type** *int*

**property oct**

Computes the octal string value of the underlying `memoryview`.

**Returns** Memory in octal string form

**Return type** *str*

**class** `ulid.api.Randomness` (*buffer: Union[bytes, bytearray, memoryview]*)

Represents the randomness portion of a ULID.

- Cryptographically secure random values.
- Last 80 bits of ULID when in binary format.
- Last 16 characters of ULID when in string format.

**property str**

Computes the string value of the randomness from the underlying `memoryview` in Base32 encoding.

**Returns** Timestamp in Base32 string form.

**Return type** *str*

**Raises** `ValueError` – if underlying `memoryview` cannot be encoded

**property bin**

Computes the binary string value of the underlying `memoryview`.

**Returns** Memory in binary string form

**Return type** *str*

**property bytes**

Computes the bytes value of the underlying `memoryview`.

**Returns** Memory in bytes form

**Return type** *bytes*

**property float**

Computes the float value of the underlying `memoryview` in big-endian byte order.

**Returns** Bytes in float form.

**Return type** *float*

**property hex**

Computes the hexadecimal string value of the underlying `memoryview`.

**Returns** Memory in hexadecimal string form

**Return type** *str*

**property int**

Computes the integer value of the underlying `memoryview` in big-endian byte order.

**Returns** Bytes in integer form.

**Return type** *int*

**property oct**

Computes the octal string value of the underlying `memoryview`.

**Returns** Memory in octal string form

**Return type** *str*

**class** `ulid.api.ULID` (*buffer: Union[bytes, bytearray, memoryview]*)

Represents a ULID.

- 128 bits in binary format.
- 26 characters in string format.
- 16 octets.
- Network byte order, big-endian, most significant bit first.

**property str**

Computes the string value of the ULID from its `memoryview` in Base32 encoding.

**Returns** ULID in Base32 string form.

**Return type** *str*

**Raises** `ValueError` – if underlying `memoryview` cannot be encoded

**timestamp** () → *ulid.ulid.Timestamp*

Creates a `Timestamp` instance that maps to the first 48 bits of this ULID.

**Returns** Timestamp from first 48 bits.

**Return type** *Timestamp*

**randomness** () → *ulid.ulid.Randomness*

Creates a `Randomness` instance that maps to the last 80 bits of this ULID.

**Returns** Timestamp from first 48 bits.

**Return type** *Timestamp*

**property uuid**

Creates a `UUID` instance of the ULID from its `bytes` representation.

**Returns** UUIDv4 from the ULID bytes

**Return type** `UUID`

**property bin**

Computes the binary string value of the underlying `memoryview`.

**Returns** Memory in binary string form

**Return type** *str*

**property bytes**

Computes the bytes value of the underlying `memoryview`.

**Returns** Memory in bytes form

**Return type** *bytes*

**property float**

Computes the float value of the underlying `memoryview` in big-endian byte order.

**Returns** Bytes in float form.

**Return type** *float*

**property hex**

Computes the hexadecimal string value of the underlying `memoryview`.

**Returns** Memory in hexadecimal string form

**Return type** *str*

**property int**

Computes the integer value of the underlying `memoryview` in big-endian byte order.

**Returns** Bytes in integer form.

**Return type** *int*

**property oct**

Computes the octal string value of the underlying `memoryview`.

**Returns** Memory in octal string form

**Return type** *str*

## 1.2 ulid/base32

Functionality for encoding/decoding ULID strings/bytes using Base32 format.

---

**Note:** This module makes the trade-off of code duplication for inline computations over multiple function calls for performance reasons. I'll check metrics in the future to see how much it helps and if it's worth it to maintain.

---

- *Base32 Documentation* <<http://www.crockford.com/wrmg/base32.html>>
- *NUlid Project* <<https://github.com/RobThree/NUlid>>

`ulid.base32`.**ENCODING**

Base32 character set. Excludes characters "I L O U".

`ulid.base32`.**DECODING**

Array that maps encoded string char byte values to enable O(1) lookups.

`ulid.base32.encode` (*value: Union[bytes, bytearray, memoryview]*) → *str*

Encode the given `bytes` instance to a `str` using Base32 encoding.

---

**Note:** You should only use this method if you've got a `bytes` instance and you are unsure of what it represents. If you know the the `_meaning_` of the `bytes` instance, you should call the `encode_*` method explicitly for better performance.

---

**Parameters** `value` (`bytes`, `bytearray`, or `memoryview`) – Bytes to encode

**Returns** Value encoded as a Base32 string

**Return type** *str*

**Raises** `ValueError` – when the value is not 6, 10, or 16 bytes long

`ulid.base32.encode_ulid` (*value*: `Union[bytes, bytearray, memoryview]`) → `str`  
Encode the given buffer to a `str` using Base32 encoding.

---

**Note:** This uses an optimized strategy from the *NULid* project for encoding ULID bytes specifically and is not meant for arbitrary encoding.

---

**Parameters** `value` (`bytes`, `bytearray`, or `memoryview`) – Bytes to encode

**Returns** Value encoded as a Base32 string

**Return type** `str`

**Raises** `ValueError` – when the value is not 16 bytes

`ulid.base32.encode_timestamp` (*timestamp*: `Union[bytes, bytearray, memoryview]`) → `str`  
Encode the given buffer to a `str` using Base32 encoding.

The given `bytes` are expected to represent the first 6 bytes of a ULID, which are a timestamp in milliseconds.

---

**Note:** This uses an optimized strategy from the *NULid* project for encoding ULID bytes specifically and is not meant for arbitrary encoding.

---

**Parameters** `timestamp` (`bytes`, `bytearray`, or `memoryview`) – Bytes to encode

**Returns** Value encoded as a Base32 string

**Return type** `str`

**Raises** `ValueError` – when the timestamp is not 6 bytes

`ulid.base32.encode_randomness` (*randomness*: `Union[bytes, bytearray, memoryview]`) → `str`  
Encode the given buffer to a `str` using Base32 encoding.

The given `bytes` are expected to represent the last 10 bytes of a ULID, which are cryptographically secure random values.

---

**Note:** This uses an optimized strategy from the *NULid* project for encoding ULID bytes specifically and is not meant for arbitrary encoding.

---

**Parameters** `randomness` (`bytes`, `bytearray`, or `memoryview`) – Bytes to encode

**Returns** Value encoded as a Base32 string

**Return type** `str`

**Raises** `ValueError` – when the randomness is not 10 bytes

`ulid.base32.decode` (*value*: `str`) → `bytes`  
Decode the given Base32 encoded `str` instance to `bytes`.

**Note:** You should only use this method if you've got a `str` instance and you are unsure of what it represents. If you know the the `_meaning_` of the `str` instance, you should call the `decode_*` method explicitly for better performance.

---

**Parameters** `value` (`str`) – String to decode

**Returns** Value decoded from Base32 string

**Return type** `bytes`

**Raises**

- **ValueError** – when value is not 10, 16, or 26 characters
- **ValueError** – when value cannot be encoded in ASCII

`ulid.base32.decode_ulid` (`value: str`) → `bytes`  
Decode the given Base32 encoded `str` instance to `bytes`.

---

**Note:** This uses an optimized strategy from the *NULid* project for decoding ULID strings specifically and is not meant for arbitrary decoding.

---

**Parameters** `value` (`str`) – String to decode

**Returns** Value decoded from Base32 string

**Return type** `bytes`

**Raises**

- **ValueError** – when value is not 26 characters
- **ValueError** – when value cannot be encoded in ASCII

`ulid.base32.decode_timestamp` (`timestamp: str`) → `bytes`  
Decode the given Base32 encoded `str` instance to `bytes`.

The given `str` are expected to represent the first 10 characters of a ULID, which are the timestamp in milliseconds.

---

**Note:** This uses an optimized strategy from the *NULid* project for decoding ULID strings specifically and is not meant for arbitrary decoding.

---

**Parameters** `timestamp` (`str`) – String to decode

**Returns** Value decoded from Base32 string

**Return type** `bytes`

**Raises**

- **ValueError** – when value is not 10 characters
- **ValueError** – when value cannot be encoded in ASCII

`ulid.base32.decode_randomness` (*randomness: str*) → bytes

Decode the given Base32 encoded `str` instance to `bytes`.

The given `str` are expected to represent the last 16 characters of a ULID, which are cryptographically secure random values.

---

**Note:** This uses an optimized strategy from the *NULid* project for decoding ULID strings specifically and is not meant for arbitrary decoding.

---

**Parameters** `randomness` (*str*) – String to decode

**Returns** Value decoded from Base32 string

**Return type** `bytes`

**Raises**

- **ValueError** – when value is not 16 characters
- **ValueError** – when value cannot be encoded in ASCII

`ulid.base32.str_to_bytes` (*value: str, expected\_length: int*) → bytes

Convert the given string to bytes and validate it is within the Base32 character set.

**Parameters**

- **value** (*str*) – String to convert to bytes
- **expected\_length** (*int*) – Expected length of the input string

**Returns** Value converted to bytes.

**Return type** `bytes`

## 1.3 ulid/hints

Contains type hint definitions across modules in the package.

`ulid.hints.Bool`

alias of `builtins.bool`

`ulid.hints.Buffer`

Type hint that defines multiple types that implement the buffer protocol that can be encoded into a Base32 string.

alias of `Union[bytes, bytearray, memoryview]`

`ulid.hints.Bytes`

alias of `builtins.bytes`

`ulid.hints.Datetime`

alias of `datetime.datetime`

`ulid.hints.Float`

alias of `builtins.float`

`ulid.hints.Int`

alias of `builtins.int`

`ulid.hints.Module`

alias of `builtins.module`



`ulid.hints.Primitive`

Type hint that defines multiple primitive types that can represent parts or full ULID.

alias of `Union[int, float, str, bytes, bytearray, memoryview]`

`ulid.hints.Str`

alias of `builtins.str`

**class** `ulid.hints.UUID` (*hex=None, bytes=None, bytes\_le=None, fields=None, int=None, version=None, \*, is\_safe=<SafeUUID.unknown: None>*)

Instances of the `UUID` class represent UUIDs as specified in RFC 4122. `UUID` objects are immutable, hashable, and usable as dictionary keys. Converting a `UUID` to a string with `str()` yields something in the form `'12345678-1234-1234-1234-123456789abc'`. The `UUID` constructor accepts five possible forms: a similar string of hexadecimal digits, or a tuple of six integer fields (with 32-bit, 16-bit, 16-bit, 8-bit, 8-bit, and 48-bit values respectively) as an argument named `'fields'`, or a string of 16 bytes (with all the integer fields in big-endian order) as an argument named `'bytes'`, or a string of 16 bytes (with the first three fields in little-endian order) as an argument named `'bytes_le'`, or a single 128-bit integer as an argument named `'int'`.

UUIDs have these read-only attributes:

**bytes** the UUID as a 16-byte string (containing the six integer fields in big-endian byte order)

**bytes\_le** the UUID as a 16-byte string (with `time_low`, `time_mid`, and `time_hi_version` in little-endian byte order)

**fields** a tuple of the six integer fields of the UUID,

which are also available as six individual attributes and two derived attributes:

`time_low` the first 32 bits of the UUID `time_mid` the next 16 bits of the UUID `time_hi_version`  
the next 16 bits of the UUID `clock_seq_hi_variant` the next 8 bits of the UUID `clock_seq_low`  
the next 8 bits of the UUID `node` the last 48 bits of the UUID

`time` the 60-bit timestamp `clock_seq` the 14-bit sequence number

`hex` the UUID as a 32-character hexadecimal string

`int` the UUID as a 128-bit integer

`urn` the UUID as a URN as specified in RFC 4122

**variant** the UUID variant (one of the constants `RESERVED_NCS`, `RFC_4122`, `RESERVED_MICROSOFT`, or `RESERVED_FUTURE`)

**version** the UUID version number (1 through 5, meaningful only when the variant is `RFC_4122`)

**is\_safe** An enum indicating whether the UUID has been generated in a way that is safe for multiprocessing applications, via `uuid_generate_time_safe(3)`.

Create a `UUID` from either a string of 32 hexadecimal digits, a string of 16 bytes as the `'bytes'` argument, a string of 16 bytes in little-endian order as the `'bytes_le'` argument, a tuple of six integers (32-bit `time_low`, 16-bit `time_mid`, 16-bit `time_hi_version`, 8-bit `clock_seq_hi_variant`, 8-bit `clock_seq_low`, 48-bit `node`) as the `'fields'` argument, or a single 128-bit integer as the `'int'` argument. When a string of hex digits is given, curly braces, hyphens, and a URN prefix are all optional. For example, these expressions all yield the same `UUID`:

```
UUID('{12345678-1234-5678-1234-567812345678}')    UUID('12345678123456781234567812345678')
UUID('urn:uuid:12345678-1234-5678-1234-567812345678')    UUID(bytes='x12x34x56x78'*4)
UUID(bytes_le='x78x56x34x12x34x12x78x56' +
      'x12x34x56x78x12x34x56x78')
```

```
UUID(fields=(0x12345678, 0x1234, 0x5678, 0x12, 0x34, 0x567812345678))
UUID(int=0x12345678123456781234567812345678)
```

Exactly one of 'hex', 'bytes', 'bytes\_le', 'fields', or 'int' must be given. The 'version' argument is optional; if given, the resulting UUID will have its variant and version set according to RFC 4122, overriding the given 'hex', 'bytes', 'bytes\_le', 'fields', or 'int'.

is\_safe is an enum exposed as an attribute on the instance. It indicates whether the UUID has been generated in a way that is safe for multiprocessing applications, via `uuid_generate_time_safe(3)`.

## 1.4 ulid/ulid

Object representation of a ULID.

**class** `ulid.ulid.Timestamp` (*buffer: Union[bytes, bytearray, memoryview]*)

Represents the timestamp portion of a ULID.

- Unix time (time since epoch) in milliseconds.
- First 48 bits of ULID when in binary format.
- First 10 characters of ULID when in string format.

**property** `str`

Computes the string value of the timestamp from the underlying `memoryview` in Base32 encoding.

**Returns** Timestamp in Base32 string form.

**Return type** `str`

**Raises** `ValueError` – if underlying `memoryview` cannot be encoded

**property** `timestamp`

Computes the Unix time (seconds since epoch) from its `memoryview`.

**Returns** Timestamp in Unix time (seconds since epoch) form.

**Return type** `float`

**property** `datetime`

Creates a `datetime` instance (assumes UTC) from the Unix time value of the timestamp with millisecond precision.

**Returns** Timestamp in datetime form.

**Return type** `datetime`

**property** `bin`

Computes the binary string value of the underlying `memoryview`.

**Returns** Memory in binary string form

**Return type** `str`

**property** `bytes`

Computes the bytes value of the underlying `memoryview`.

**Returns** Memory in bytes form

**Return type** `bytes`

**property** `float`

Computes the float value of the underlying `memoryview` in big-endian byte order.

**Returns** Bytes in float form.

**Return type** `float`

**property hex**

Computes the hexadecimal string value of the underlying `memoryview`.

**Returns** Memory in hexadecimal string form

**Return type** `str`

**property int**

Computes the integer value of the underlying `memoryview` in big-endian byte order.

**Returns** Bytes in integer form.

**Return type** `int`

**property oct**

Computes the octal string value of the underlying `memoryview`.

**Returns** Memory in octal string form

**Return type** `str`

**class** `ulid.ulid.Randomness` (*buffer: Union[bytes, bytearray, memoryview]*)

Represents the randomness portion of a ULID.

- Cryptographically secure random values.
- Last 80 bits of ULID when in binary format.
- Last 16 characters of ULID when in string format.

**property str**

Computes the string value of the randomness from the underlying `memoryview` in Base32 encoding.

**Returns** Timestamp in Base32 string form.

**Return type** `str`

**Raises** `ValueError` – if underlying `memoryview` cannot be encoded

**property bin**

Computes the binary string value of the underlying `memoryview`.

**Returns** Memory in binary string form

**Return type** `str`

**property bytes**

Computes the bytes value of the underlying `memoryview`.

**Returns** Memory in bytes form

**Return type** `bytes`

**property float**

Computes the float value of the underlying `memoryview` in big-endian byte order.

**Returns** Bytes in float form.

**Return type** `float`

**property hex**

Computes the hexadecimal string value of the underlying `memoryview`.

**Returns** Memory in hexadecimal string form

**Return type** `str`

**property int**

Computes the integer value of the underlying `memoryview` in big-endian byte order.

**Returns** Bytes in integer form.

**Return type** `int`

**property oct**

Computes the octal string value of the underlying `memoryview`.

**Returns** Memory in octal string form

**Return type** `str`

**class** `ulid.ulid.ULID` (*buffer: Union[bytes, bytearray, memoryview]*)

Represents a ULID.

- 128 bits in binary format.
- 26 characters in string format.
- 16 octets.
- Network byte order, big-endian, most significant bit first.

**property str**

Computes the string value of the ULID from its `memoryview` in Base32 encoding.

**Returns** ULID in Base32 string form.

**Return type** `str`

**Raises** `ValueError` – if underlying `memoryview` cannot be encoded

**timestamp** () → `ulid.ulid.Timestamp`

Creates a `Timestamp` instance that maps to the first 48 bits of this ULID.

**Returns** Timestamp from first 48 bits.

**Return type** `Timestamp`

**randomness** () → `ulid.ulid.Randomness`

Creates a `Randomness` instance that maps to the last 80 bits of this ULID.

**Returns** Timestamp from first 48 bits.

**Return type** `Timestamp`

**property uuid**

Creates a `UUID` instance of the ULID from its `bytes` representation.

**Returns** UUIDv4 from the ULID bytes

**Return type** `UUID`

**property bin**

Computes the binary string value of the underlying `memoryview`.

**Returns** Memory in binary string form

**Return type** `str`

**property bytes**

Computes the bytes value of the underlying `memoryview`.

**Returns** Memory in bytes form

**Return type** `bytes`

**property float**

Computes the float value of the underlying `memoryview` in big-endian byte order.

**Returns** Bytes in float form.

**Return type** `float`

**property hex**

Computes the hexadecimal string value of the underlying `memoryview`.

**Returns** Memory in hexadecimal string form

**Return type** `str`

**property int**

Computes the integer value of the underlying `memoryview` in big-endian byte order.

**Returns** Bytes in integer form.

**Return type** `int`

**property oct**

Computes the octal string value of the underlying `memoryview`.

**Returns** Memory in octal string form

**Return type** `str`



## PYTHON MODULE INDEX

### U

ulid.api, 3  
ulid.base32, 9  
ulid.hints, 12  
ulid.ulid, 14





**B**

bin() (*ulid.api.Randomness property*), 7  
 bin() (*ulid.api.Timestamp property*), 6  
 bin() (*ulid.api.ULID property*), 8  
 bin() (*ulid.ulid.Randomness property*), 15  
 bin() (*ulid.ulid.Timestamp property*), 14  
 bin() (*ulid.ulid.ULID property*), 16  
 Bool (*in module ulid.hints*), 12  
 Buffer (*in module ulid.hints*), 12  
 Bytes (*in module ulid.hints*), 12  
 bytes() (*ulid.api.Randomness property*), 7  
 bytes() (*ulid.api.Timestamp property*), 6  
 bytes() (*ulid.api.ULID property*), 8  
 bytes() (*ulid.ulid.Randomness property*), 15  
 bytes() (*ulid.ulid.Timestamp property*), 14  
 bytes() (*ulid.ulid.ULID property*), 16

**C**

create() (*in module ulid.api*), 3

**D**

Datetime (*in module ulid.hints*), 12  
 datetime() (*ulid.api.Timestamp property*), 6  
 datetime() (*ulid.ulid.Timestamp property*), 14  
 decode() (*in module ulid.base32*), 10  
 decode\_randomness() (*in module ulid.base32*), 11  
 decode\_timestamp() (*in module ulid.base32*), 11  
 decode\_ulid() (*in module ulid.base32*), 11  
 DECODING (*in module ulid.base32*), 9

**E**

encode() (*in module ulid.base32*), 9  
 encode\_randomness() (*in module ulid.base32*), 10  
 encode\_timestamp() (*in module ulid.base32*), 10  
 encode\_ulid() (*in module ulid.base32*), 10  
 ENCODING (*in module ulid.base32*), 9

**F**

Float (*in module ulid.hints*), 12  
 float() (*ulid.api.Randomness property*), 7  
 float() (*ulid.api.Timestamp property*), 6  
 float() (*ulid.api.ULID property*), 9

float() (*ulid.ulid.Randomness property*), 15  
 float() (*ulid.ulid.Timestamp property*), 14  
 float() (*ulid.ulid.ULID property*), 16  
 from\_bytes() (*in module ulid.api*), 4  
 from\_int() (*in module ulid.api*), 4  
 from\_randomness() (*in module ulid.api*), 5  
 from\_str() (*in module ulid.api*), 4  
 from\_timestamp() (*in module ulid.api*), 5  
 from\_uuid() (*in module ulid.api*), 5

**H**

hex() (*ulid.api.Randomness property*), 7  
 hex() (*ulid.api.Timestamp property*), 7  
 hex() (*ulid.api.ULID property*), 9  
 hex() (*ulid.ulid.Randomness property*), 15  
 hex() (*ulid.ulid.Timestamp property*), 14  
 hex() (*ulid.ulid.ULID property*), 17

**I**

Int (*in module ulid.hints*), 12  
 int() (*ulid.api.Randomness property*), 8  
 int() (*ulid.api.Timestamp property*), 7  
 int() (*ulid.api.ULID property*), 9  
 int() (*ulid.ulid.Randomness property*), 15  
 int() (*ulid.ulid.Timestamp property*), 15  
 int() (*ulid.ulid.ULID property*), 17

**M**

module  
     ulid.api, 3  
     ulid.base32, 9  
     ulid.hints, 12  
     ulid.ulid, 14  
 Module (*in module ulid.hints*), 12

**N**

new() (*in module ulid.api*), 3

**O**

oct() (*ulid.api.Randomness property*), 8  
 oct() (*ulid.api.Timestamp property*), 7  
 oct() (*ulid.api.ULID property*), 9

oct () (*ulid.ulid.Randomness property*), 16  
oct () (*ulid.ulid.Timestamp property*), 15  
oct () (*ulid.ulid.ULID property*), 17

## P

parse () (*in module ulid.api*), 3  
Primitive (*in module ulid.hints*), 12

## R

Randomness (*class in ulid.api*), 7  
Randomness (*class in ulid.ulid*), 15  
randomness () (*ulid.api.ULID method*), 8  
randomness () (*ulid.ulid.ULID method*), 16

## S

Str (*in module ulid.hints*), 13  
str () (*ulid.api.Randomness property*), 7  
str () (*ulid.api.Timestamp property*), 6  
str () (*ulid.api.ULID property*), 8  
str () (*ulid.ulid.Randomness property*), 15  
str () (*ulid.ulid.Timestamp property*), 14  
str () (*ulid.ulid.ULID property*), 16  
str\_to\_bytes () (*in module ulid.base32*), 12

## T

Timestamp (*class in ulid.api*), 6  
Timestamp (*class in ulid.ulid*), 14  
timestamp () (*ulid.api.Timestamp property*), 6  
timestamp () (*ulid.api.ULID method*), 8  
timestamp () (*ulid.ulid.Timestamp property*), 14  
timestamp () (*ulid.ulid.ULID method*), 16

## U

ULID (*class in ulid.api*), 8  
ULID (*class in ulid.ulid*), 16  
ulid.api  
    module, 3  
ulid.base32  
    module, 9  
ulid.hints  
    module, 12  
ulid.ulid  
    module, 14  
UUID (*class in ulid.hints*), 13  
uuid () (*ulid.api.ULID property*), 8  
uuid () (*ulid.ulid.ULID property*), 16